Project Kilo: A Mobile Map Service Personal Report

Michael J A Smith

February 26, 2004

1 Introduction

This document details my personal view of the group project work undertaken by Project Kilo. I will discuss the contributions that I made over the course of the project, in addition to those made by other group members. Included are some samples of my work, as detailed below.

2 Personal Contributions

From the outset of the project, my primary responsibility was for the backend of the server, which carried through the design, implementation, and testing phases.

After the first project meetings, I designed and formalised the specification of the XML datatype to be used by the server, on the basis of our discussions as a group. This resulted in a technical document [attached], and the DTD (document type definition) of the file format. Similarly, I spent a considerable time formalising the technical specification of the server, culminating in the UML diagram detailing the server classes (which I created using ArgoUML). This was complemented by two collaboration diagrams, showing the execution of both WAP and Java client requests on the server.

During the implementation phase, I began by creating the class stubs from the UML, and implementing the core data structure classes of the server (Edge, Node, Road, Subnode, TransportType, Position, Location, MetaLocation, Category). I then proceeded by coding the parser for the data format, in the MapParser class. This takes reads in the XML file using the DOM parser, and converts it into the internal data structure. For the Map class, I coded an implementation of the SplayTree data structure in order to perform efficient resolution of Positions into Nodes.

The next stage in the implementation was the routing. I coded the Router class, implementing the A-star algorithm in order to make it efficient. I reused the SplayTree class (which is thread safe) in order to implement a priority queue. On testing the speed of this code, I found that it took in the order of 4 to 8 ms to calculate a route over a relatively large dataset.

After having written this code, I installed and configured Apache Tomcat on my machine for use as a test server. It was found that Gavin could not finish his classes in time, so I took the Server class from him to code. When Gavin finished coding the client and the servlet, I debugged his code with him, and integrated it with the server code already written. Similarly, I worked with Alex to debug the WMLGenerator, and integrate it with the system.

In the final phase of the project, I spent some time implementing new features, and ironing out bugs in the existing code. In particular, I implemented the facility to click on the map in the Java client (by resolving a Position on the server to the nearest Node), and the facility to route from MetaLocations and Categories (so that for example you can ask for a route to 'places to eat').

I spent some time working on the client, with regard to configuring the .jar for the applet, and the web page itself. Furthermore, I helped Alex to debug the modifications he had made to the WMLGenerator class, and integrated it into the system.

With regard to testing, I performed module-level tests on the classes, as I wrote them, to ensure that they executed successfully, and with sensible output. System wide testing was performed with the actual map data, whereby I was able to use the data to locate bugs in the code, and conversely the correct operation of the code to locate the data entry errors.

3 Contributions Of Other Group Members

Below is a summary of the work done by the other group members:

- Alex - coded most of the WMLGenerator class. He worked on the design of the WML page format, and implemented a memory of selections by sending strings of parameters with each page request.

- Adam - did a great job with the instruction generation in the NodeRoute class. He made substantial contributions during the design phase with regard to discussing, modifying, and finalising features in the design. He also coded the ClientDataGenerator, and the data classes for the client.

- James - has done a fantastic job with the dataset. He created a new data format with which to enter map data more efficiently and more quickly, and a perl script to convert it to XML. He then went about collecting data from the university map, and converting it into the required format. In the design phase, James created a large number of documents, detailing the technical specification of the project.

- Gavin - was the main coder of the client classes, and the client-server interface. He created a prototype client during the design phase, which was coded and modified throughout the project. He wrote the MapServlet, Communicator and Communication classes to send data via HTTP to and from the client. He also worked on the migration of the server from my machine to the Clare College server.

- Alaric - did a great job coding the rendering classes. He decided to implement it using SVG (scalable vector graphics) on the server side, for much more flexibility, which has yielded pleasing results. He coded the Renderer, RouteRenderer, MapRenderer, and MapImage classes. He gave substantial input during the design phase, and organised accounts on the Clare Server.

- Chris - was extremely efficient and organised as our secretary. He wrote up all the meeting notes in great detail, so that we had a record of all the points that were decided upon - this was particularly useful during the design phase. He also produced the user documentation.

- Alexia - produced three management documents for the review meetings.

4 Examples Of Work

This section lists a number of documents and code fragments from the classes I have coded. These are all included at the end of this document

- XML Document Specification - this details the format of the XML map data, and includes the DTD that it must conform to.

- Server UML Class Diagram - this shows the structure of the classes on the server.

- Collaboration Diagrams - these detail the operations performed when a web user connects via the Java applet, and when a mobile user connects via WAP.

- MapParser.java - the exerpt lists part of the `getMap()` method that creates and links the Edge objects from the XML data.
- Router.java - the exerpt lists the `route()` method, which performs the actual A-star algorithm. This is the core of the Router.
- SplayTree.java - the exerpt lists the `splay()` and `rotate()` methods, which provide the core of the functionality of a splay tree.
- Map.java - the exerpt lists the `getNode()` and `hash()` methods, which perform Position to Node resolution using a SplayTree.

5 Conclusions

The aim of Project Kilo was to produce a mobile map service through which advertisers could attract customers with a location finding service. I feel that we have achieved much more than this. The implemented features allow our product to be used in addition for generating travel instructions, finding routes in general between points on the map, and for generating attractive maps. The service could easily be scaled to other cities, and even to larger scale road maps between towns.

It was a great opportunity to be confronted with so many technologies to use, and I feel that I have gained a lot from that. In particular, I learnt to use XML, WML, and Java Servlet technology. There is a great sense of satisfaction in seeing a completed product that can be of great use to all manner of people, and in this way, the project has turned out to be much more than just a software engineering exercise.